Self-Regulated Priority based Round Robin Scheduling Algorithm

Md.Gulzar¹, Lt. K. Ravindra Babu², A.Vinaya Babu³, Ph.D, S.Udaya Kumar⁴, Ph.D
¹PGStudent, Vagdevicollege of engineering, Warangal, ²Associate Professor in CSE, KITS(S)
³Principal, JNTUCE, JNTUH, Kukatpally, Hyd, ⁴Principal, GCET, Cheeryala, Keesara, Hyderabad,

ABSTRACT
The main aim of this paper is to present a new scheduling algorithm even though there exists good scheduling algorithms. Each scheduling algorithm is having its own merits and demerits. The proposed algorithm overcomes the demerits of existing scheduling algorithms like high average waiting time, high average turnaround time, low throughput, high number of context switches. The proposed algorithm is a preemptive algorithm which takes a time quantum to execute the processes like round robin scheduling algorithm. But the time quantum is calculated automatically depending on the average of all burst times. And to avoid the problem of starvation high priority should be assigned for short process.

General Terms
CPU Scheduling Algorithms, Round Robin Scheduling, Self-Regulated Priority based Round Robin Scheduling.

Keywords
Average waiting time, average turnaround time, context switches.

1. INTRODUCTION
CPU scheduling is the basic of multiprogrammed operating systems. The operating system switches the CPU among processes to make the system more productive. To achieve this CPU must be as busy as possible; the CPU time should not be wasted because it affects the throughput of the system. In general it can be said that CPU must execute some process all the time [6].

To increase the performance the processes should be scheduled very efficiently. The scheduler deals with this. Whenever a CPU is idle it fetches a process from memory and the dispatcher gives the control over CPU to the selected process by scheduler. The scheduler selects the process from memory by using CPU scheduling algorithms [7].

1.1 Scheduling Criteria
Different CPU scheduling algorithms are exists, but each scheduling algorithm has properties; the choice of selecting a scheduling algorithm may favor one class of processor over another. For comparing scheduling algorithms many scheduling criteria have been suggested. The criteria are as follows [8-13]

1.1.1 CPU utilization
The CPU must be as busy as possible, its utilization ranges from 0 to 100 percent.

1.1.2 Throughput
Throughput can be given as the number of processes that completes the execution per time unit.

1.1.3 Turnaround time
Turnaround time of a process can be given as the interval from the time of submission of a process to the time of completion.

1.1.4 Waiting time
The waiting time can be given as the time a process waits in the ready queue to acquire the processor.

1.1.5 Response time
Response time is the interval from the time of submission of the request to time of first response produced.

1.2 Scheduling Objectives
It can be concluded that a good scheduling algorithm for real time and time sharing system must possess following characteristics[1]

- Minimum context switches.
- Maximum CPU utilization.
- Maximum throughput.
- Minimum turnaround time.
- Minimum waiting time.
- Minimum response time.
2. SCHEDULING ALGORITHMS
Scheduling Algorithms deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU. There exist various processes scheduling algorithm like First Come First Serve (FCFS), Shortest Job First (SJF),Priority Scheduling Algorithm, Round Robin Scheduling Algorithm, Multilevel Queue Scheduling, and Multilevel Feedback Queue Scheduling [3].

- First come first serve scheduling algorithm is based on FIFO. And it is non-preemptive in nature.
- Shortest job first scheduling algorithm deals with executing shortest process first, it is both preemptive and non-preemptive scheduling.
- Priority scheduling is based on priority of a process where high priority process executes first, and it is preemptive scheduling.
- The Round robin scheduling algorithm is preemptive scheduling algorithm where every process executes up to a time quantum in FIFO order.
- In Multilevel queue scheduling the processes are permanently assigned to a queue on entry to the system.
- Multilevel feedback queue deals with those processes that move between queues.

The proposed algorithm for process scheduling is based on round robin scheduling. This will overcome the drawbacks of round robin scheduling algorithm. The name given to it is “Self-Regulated Priority Based Round Robin (SRPB RR) Scheduling Algorithm”

2.1 ROUND ROBIN SCHEDULING ALGORITHM
Round Robin scheduling algorithm is designed for time sharing systems. It is similar to FCFS scheduling but preemption is added to switch between processes. A small unit of time called as time quantum or time slice is defined. A time quantum is generally ranges from 10 to 100 milliseconds. The processes are stored in ready queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of 1 time quantum. If the process burst time is bigger than time quantum, then process execution will preempted and the process moves back at the end of ready queue and the scheduler selects the next process from ready queue and allocate the CPU to that process. This process continues till the ready queue becomes empty [4].

2.2 DRAWBACKS
- If small time quantum is given the number of context switches will increase which affects the execution time of a process. If big time quantum is given it works in the same manner as first come first serve algorithm [2].
- The shortest burst time processes may have to wait for a long time which causes starvation.
- Can’t be used for real time applications which need faster execution [5].

3. PROPOSED ALGORITHM
In the proposed algorithm we consider at the beginning the ready queue is empty. All the processes enter into the ready queue at 0 millisecond. The proposed algorithm works in following steps

1. First we calculate the time quantum as average of all CPU burst time of processes in ready queue.

Time Quantum (T.Q) = \[\text{Sum of CPU burst times of process} \over \text{Number of processes in ready Queue}\]

2. Sort the processes in ready queue in ascending order and assign high priority for short process.
3. Allocate the CPU to high priority process first and if its burst time is bigger than T.Q preempt the execution and give the CPU to next higher priority Process.
4. Repeat step 2&3 till ready Queue becomes empty.

3.1 Pseudo Code for Proposed Algorithm
Step 1: begin
Step 2: initialize q=0,count=1, awt=0, att=0
Step 3: read number of processes
Step 4: for each process i=1 to
4.1: read burst time
4.2: End for
Step 5: calculate time quantum
T.Q = avg of all process burst time
Step 6: sort the processes in ascending order of their burst time, give high priority for short process
Step 7: while count = 1
7.1: count=0;
7.2: for each process i=1 to n
7.2.1: if bt[i]>tq then go to 7.2.1.1 else go to 7.2.2
7.2.1.1: bt[i]=bt[i]-tq;
7.2.1.2: increment cst and count=1;
7.2.2: if bt[i] !=0 then goto 7.2.2.1 else go to 7
7.2.2.1: bt[i]=0;
7.2.2.2: increment cst and count=1;
end for and while
Step 8: for each process i=1 to n find waiting time, turnaround time, number of context switches, AWT and ATT.
Step 9: Print the result
Step 10: Stop.
3.2 Flow Chart

![Flow Chart](chart.png)

4. EXPERIMENTAL ANALYSIS
To evaluate the performance of proposed algorithm two examples are considered. These examples are solved with general round robin scheduling algorithm and with proposed self-regulated priority based round robin scheduling algorithm. And the analysis on these algorithms is done by comparing their results like average waiting time, average turnaround time and number of context switches.
4.1 Example1
Consider initially the ready queue is empty, and five processes arrived in ready queue at 0 ms, and their burst times are given as follows:

<table>
<thead>
<tr>
<th>PROCESS</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>15</td>
</tr>
<tr>
<td>P2</td>
<td>20</td>
</tr>
<tr>
<td>P3</td>
<td>5</td>
</tr>
<tr>
<td>P4</td>
<td>25</td>
</tr>
<tr>
<td>P5</td>
<td>10</td>
</tr>
</tbody>
</table>

4.1.1 Scheduling with round robin algorithm
In round robin scheduling the scheduler selects a process from ready queue in First in first out order (FIFO). And each process executes up to a time value of time quantum. Consider the time quantum is 10. To show the process scheduling Gantt chart is used.

**Fig 2: Gantt chart for example1 with RR scheduling**

From the above Gantt chart the results of round robin scheduling are as follows:
The average waiting time is 36.00
The average turnaround time is 51.00
The number of context switches is 8

4.1.2 Scheduling with proposed algorithm
In the proposed algorithm the time quantum is the average of all burst times of the processes in the ready queue.

\[ T.Q = \frac{15+20+5+25+10}{5} = 15 \]

And the processes in the ready queue will be sorted in ascending order, and high priority is assigned for shortest process. The higher priority or shortest burst time process executes first. The Gantt chart for this algorithm which shows process scheduling is as follows:

**Fig 3: Gantt chart for example1 with SRPB RR Scheduling**

From the above Gantt chart the results of proposed algorithm is as follows:
The average waiting time is 23.00
The average turnaround time is 38.00
The number of context switches is 6.

Proposed algorithm is implemented in c programming language. When program is compiled with above input following output is generated:

**Fig 4: Result of example1 with SRPB RR scheduling**

4.1.3Analysis
Following table shows the analysis of results in above two algorithms:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average waiting time (awt)</th>
<th>Average turnaround time (att)</th>
<th>Number of context switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>36.00</td>
<td>51.00</td>
<td>8</td>
</tr>
<tr>
<td>SRPB RR</td>
<td>23.00</td>
<td>38.00</td>
<td>6</td>
</tr>
</tbody>
</table>

Following chart as shown in figure5 graphically represents above analysis.
4.2 Example 2

Consider the ready queue is having four processes as follows.

<table>
<thead>
<tr>
<th>PROCESS</th>
<th>BURST TIMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>3</td>
</tr>
<tr>
<td>P2</td>
<td>8</td>
</tr>
<tr>
<td>P3</td>
<td>5</td>
</tr>
<tr>
<td>P4</td>
<td>4</td>
</tr>
</tbody>
</table>

4.2.1 Scheduling with round robin algorithm

Consider the time quantum is 5. The following figure 6 is the Gantt chart which shows the scheduling of above processes.

From the above Gantt chart the results of round robin algorithm is as follows

- The average waiting time is 8.25.
- The average turnaround time is 13.25.
- The number of context switches is 4.

4.2.2 Scheduling with proposed algorithm

In proposed algorithm the time quantum for above processes is

\[ T.Q = \frac{3+8+5+4}{4} = 5 \]

The processes are sorted in ascending order of their burst times. Following figure 7 is the Gantt chart which shows the scheduling of above processes with proposed algorithm.

From the above Gantt chart the results of proposed algorithm is as follows

- The average waiting time is 5.75.
- The average turnaround time is 10.50.
- The number of context switches is 4.

The generated output for above example is shown in figure 8.

4.2.3 Analysis

Following table shows the analysis of results in above two algorithms.
Table 2. Analysis of two algorithms for example 2

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average waiting time (att)</th>
<th>Average turnaround time (awt)</th>
<th>Number of context switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>8.25</td>
<td>13.25</td>
<td>4</td>
</tr>
<tr>
<td>SRPB RR</td>
<td>5.5</td>
<td>10.5</td>
<td>4</td>
</tr>
</tbody>
</table>

Following chart as shown in figure 9 graphically represents above analysis.

Fig 9: Analysis of example 2 in two algorithms

From the above chart it can be seen that there is same number of context switches, it is because the time quantum which is taken is same i.e 5 in both algorithms. But the average waiting time and average turnaround time is decreased with proposed algorithm.

5. CONCLUSION AND FUTURE WORK

With proposed algorithm the drawbacks of round robin scheduling are eliminated. It is shown by comparing the performance of two algorithms. And the problem of starvation is eliminated by allowing the shortest burst time process to execute first. The only limitation in proposed algorithm is the consideration that all processes will arrive in ready queue at once i.e at 0ms. It can be eliminated in future if proper work is done on this.

6. ACKNOWLEDGEMENTS

First author would like to thank other authors for their excellent guidance.

7. REFERENCES


